```
In [ ]: import math
        import numpy as np
```

Define Engine Characteristics

```
In [ ]: # Provided Engine Characteristics
        mechanical_eff = 0.99
        gamma = 1.4
        gamma_g = 1.33333
        c_p_air = 1.005
        c_p_gas = 1.148
        # ENGINE INLET
        inlet_loss = 0.01            # Inlet pressure loss
        # ------------------------------------------------------------------------------
        # Compressor
        m_dot = 5.21                 # [lb/s]
        lpc_pr = 4                   # LPC PR
        lpc_eff = 0.865              # LPC Target Efficiency
        hpc_pr = 3                   # HPC PR
        hpc_eff = 0.855              # HPC Target Efficiency
        hpc_bleed_air = 0.09         # HPC Bleed Air (exit)
        # ------------------------------------------------------------------------------
        # Combustor
        AFR = 0.02                   # Fuel to Air Ratio or Air Fuel Ratio (AFR)
        FHV = 40007.2                # [kJ/kg] Fuel Heating Value (FHV)
        combustor_eff = 0.99
        combustor_loss = 0.018       # Combustor pressure loss
        RTDF = 0.05                  # Radial Temperature Distribution Factor (RTDF)
        # ------------------------------------------------------------------------------
        # Turbine
        hpt_eff = 0.83               # HPT Target Efficiency->Given range 0.83-0.85
        hpt_vane_cooling = 0.03      # HPT Vane Cooling Air
        hpt_disk_cooling = 0.165     # HPT Disk Cooling Air
        lpt_eff = 0.91               # LPT Target Efficiency
        hpt_vane_cooling = 0.011     # LPT Vane Cooling Air
        ITD_loss = 0.006             # ITD? Loss
        pt_eff = 0.92                # PT Target Efficiency
        pt_disk_cooling = 0.0125     # PT disk cooling air
        # ------------------------------------------------------------------------------
        # Exhaust
        exhaust_loss = 0.02          # Exhaust Loss
        exhaust_mach = 0.15          # Exhaust Mach Number
```

Functions for calculations

```
In [ ]: def calc_turbine_pressure(P_i, T_03, T_04, eta_turbine):
            """
            This function calculation the exit pressure of a turbine using the isentropic efficiency.
            Inputs: P_i, T_03, T_04, eta_turbine
            Outputs: P_f

            """
            temp_val_1 = 1.33333/(1.33333 - 1)
            temp_val_2 = 1 - ((T_03 - T_04) / (eta_turbine * T_03))
            P_f = P_i * (temp_val_2**temp_val_1)
            print(P_f)
            return P_f

        def calc_psi(c_p, delta_T_0_s, U):
            """
            This function calculates the blade loading coefficient

            ___________
            Input: c_p, delta_T_0_s, U
            """
            psi = (2 * c_p * delta_T_0_s)/(U**2)
            print("psi = ", psi)
            return psi

        def calc_lambda(alpha, phi, psi):
            """
            This function calculates the degree of reaction

            ___________
            Input: Swirl angle "alpha", flow coefficient "phi", blade loading coefficient "psi"

            ___________
            Output: Returns the value of the degree of coefficient and the tan_beta_3
            """
            alpha = math.radians(alpha)
            tan_beta_3 = math.tan(alpha) + (1/phi)
            lambda_val = (4 * tan_beta_3 * phi - psi)/4
            print("lambda = ", lambda_val)
            return lambda_val

        def calc_T_static(T_total, M):
            T_static = T_total/(1 + 0.15*(M**2))
```

```python
    return T_static


def calc_area(T_static, p_static, C_a, n):
    """
    This function calculate air properties and also the area from mass flow rate
    ___________
    Input: Total Temperature "T_01", Total Pressure "p_01", Flow velocity "c", axial component of the velocity "c_a", station
    ___________
    Output: A plot showing stability in real and imaginary axis.
    """
    rho_ = p_static/(R*T_static)
    A = mass_flow_rate/(rho_* C_a)/1000
    print("rho_",n, " = ", rho_ * 1000)
    print("A_",n, " = ", A)
    return A

def calc_height(A, r_m, n):
    """
    This calculated geometric values related to the cross section of the turbine
    ___________
    Input: area "A", station number "n"
    ___________
    Output: A plot showing stability in real and imaginary axis.
    """
    h = (revs * A)/(U)
    rtrm = (r_m + 0.5*h)/(r_m - 0.5*h)
    print("h_",n,"=", h)
    print("rtrm_",n,"=", rtrm)
    return h

def calc_freevortex_nozzle(r_m, r_r, r_t, alpha):
    tan_alpha_r_2 = (r_m / r_r) * math.tan(math.radians(alpha))
    tan_alpha_t_2 = (r_m / r_t) * math.tan(math.radians(alpha))

    alpha_2_r_fv = np.arctan(tan_alpha_r_2)
    alpha_2_t_fv = np.arctan(tan_alpha_t_2)
    print("nozzle alpha values: root, tip")
    print(math.degrees(alpha_2_r_fv), math.degrees(alpha_2_t_fv))

    beta_2_r_fv = math.degrees(np.arctan(tan_alpha_r_2 - ((r_m / r_r) * (1/phi))))
    beta_2_t_fv = math.degrees(np.arctan(tan_alpha_t_2 - ((r_m / r_t) * (1/phi))))
    print("nozzle beta values: root, tip")
    print(beta_2_r_fv, beta_2_t_fv)

    return math.degrees(alpha_2_r_fv), beta_2_r_fv

def calc_freevortex_rotor(r_m, r_r, r_t, alpha):
    tan_alpha_r_3 = (r_m / r_r) * math.tan(math.radians(alpha))
    tan_alpha_t_3 = (r_m / r_t) * math.tan(math.radians(alpha))

    alpha_3_r_fv = math.degrees(np.arctan(tan_alpha_r_3))
    alpha_3_t_fv = math.degrees(np.arctan(tan_alpha_t_3))
    print("rotor alpha values: root, tip")
    print(alpha_3_r_fv, alpha_3_t_fv)

    beta_3_r_fv = math.degrees(np.arctan(tan_alpha_r_3 + ((r_m / r_r) * (1/phi))))
    beta_3_t_fv = math.degrees(np.arctan(tan_alpha_t_3 + ((r_m / r_t) * (1/phi))))
    print("rotor beta values: root, tip")
    print(abs(beta_3_r_fv), abs(beta_3_t_fv))

    return alpha_3_r_fv, beta_3_r_fv
```

Cycle Calculations

Inlet

In [ ]:

```python
# =============================================================================
# CYCLE CALCULATIONS
# =============================================================================
# Conditions at the Inlet
P_a = P_0 = 101.325        # [kPa]
T_a = T_0 = 296.483        # [K]

T_01 = T_0
P_01 = 0.99*P_0
print(P_01)
```

```
100.31175
```

Low Pressure Compressor

In [ ]:

```python
# LPC Calculations

P_02 = lpc_pr * P_01         # Using provided pressure ratio
T_02 = T_01 + (T_01/lpc_eff * (lpc_pr**(0.285714) - 1))
```

```
W_lpc = m_dot * c_p_air * (T_02 - T_01)         # [kJ/kg]
print(P_02, T_02, W_lpc/m_dot)
```

```
401.247 463.059728696695 167.40961234017846
```

High Pressure Compressor

In [ ]:
```
# HPC Calculations

P_03 = hpc_pr * P_02
T_03 = T_02 + (T_02/hpc_eff * (hpc_pr**(0.285714) - 1))
W_hpc = m_dot * c_p_air * (T_03 - T_02)         # [kJ/kg]
print(P_03, T_03, W_hpc/m_dot)
```

```
1203.741 662.7644873827279 200.70328247946304
```

Combustion Chamber

In [ ]:
```
# Combustion Calculations

m_air = 0.91 * m_dot       # Mass flow into combustor/turbine | See next line:
# Turbine cooling air percentage can be considered as percent
# flow of turbine inlet flow.
P_04 = P_03 * 0.982
print(P_04)
T_04 = ((c_p_air * T_03) + (AFR * FHV * combustor_eff))/((1 + AFR) * c_p_gas)
print(T_04)
```

```
1182.073662
1245.3208220773054
```

High Pressure Turbine

In [ ]:
```
# HPT Calculations

m_turbine = m_air + (0.02 * m_air)      # Includes the mass of the fuel as well
m_cool_vane_hpt = 0.03 * m_turbine          # HPT Vane Cooling Air MFR
m_cool_disc_hpt = 0.0165 * m_turbine        # HPT Disk Cooling Air MFR

# Calculation of cooling after stator
T_hpt_after_vane = ((m_turbine * c_p_gas * T_04) + (m_cool_vane_hpt * c_p_air * T_03)) / (c_p_gas * (m_turbine + m_cool_vane_h
print(T_hpt_after_vane)
# Calculation after rotor but before disc cooling
T_hpt_required_energy = T_04 - ((1.01 * W_hpc) / ((m_turbine + m_cool_vane_hpt) * c_p_gas))

# Calculation after disc cooling
T_hpt_after_rotor = (((m_turbine + m_cool_vane_hpt) * c_p_gas * T_hpt_required_energy) + (m_cool_disc_hpt * c_p_air * T_03)) /
T_05 = T_hpt_after_rotor
print(T_05)
# Pressure
P_05 = calc_turbine_pressure(P_04, T_04, T_05, 0.84)
```

```
1225.9485919279928
1053.0511295978204
524.5947533586902
```

Low Pressure Turbine

In [ ]:
```
# LPT Calculations

m_turbine_lpt = m_turbine + m_cool_vane_hpt + m_cool_disc_hpt
m_cool_disc_lpt = 0.011 * m_turbine

# Calculation of work done
T_lpt_required_energy = T_05 - ((1.01 * W_lpc) / (m_turbine_lpt * c_p_gas))
print(T_lpt_required_energy)
# Calculation after disc cooling
T_lpt_after_rotor = ((m_turbine_lpt * c_p_gas * T_lpt_required_energy) + (m_cool_disc_lpt * c_p_air * T_03)) / (c_p_gas * (m_t
T_06 = T_lpt_after_rotor
print(T_06)
# Pressure
P_06 = calc_turbine_pressure(P_05, T_05, T_06, lpc_eff)
```

```
901.4232437286498
898.0819933981012
248.8071191334912
```

Exhaust

In [ ]:
```
# Exhaust Calculations
P_08 = P_0 * ((1 + ((gamma_g-1)/(2)) * exhaust_mach**2 )**(gamma_g/(gamma_g-1)))
print(P_08)
P_07 = 1.02 * P_08      # Power Turbine Exit Total Pressure
print(P_07)
```

```
102.85344186914772
104.91051070653067
```

Power Pressure Turbine

In [ ]:
```
# Power Turbine Calculations
m_turbine_pt = m_turbine_lpt + m_cool_disc_lpt
m_cool_disc_pt = 0.0125 * m_turbine
# Calculation of work done
P_06_PT = (1 - ITD_loss) * P_06
pt_pr = P_06_PT / P_07
T_pt_required_energy = T_06 - (pt_eff * T_06 * (1 - (1 / pt_pr)**((gamma_g - 1)/gamma_g)))

# PT Temperature after disc cooling
T_pt_after_rotor = ((m_turbine_pt * c_p_gas * T_pt_required_energy) + (m_cool_disc_pt * c_p_air * T_03)) / (c_p_gas * (m_turbi
T_07 = T_pt_after_rotor         # Total temperature at power turbine exit

print(P_06_PT, T_05, T_06, T_pt_required_energy,T_07)
```

```
247.31427641869024 1053.0511295978204 898.0819933981012 738.6486734668399 736.7977261944576
```

Work and SFC Calculations

In [ ]:
```
# Calculation of work
W_pt = c_p_gas * (T_06 - T_pt_required_energy) * 0.99
SFC = (3600 * AFR) / W_pt
print(W_pt, SFC)
```

```
181.19915676827705 0.397352842497362
```